- **What system monitoring tools are available and how are they used?**

The *sine qua non* of system monitoring is to obtain measurements with minimal impact on the system. There are several tools available to ECS that, when used correctly, meet this criterion and provide the operations staff with useful information. These tools are described in the subsequent subsections. It should be kept in mind when going through the subsections that using too many tools simultaneously or sampling too often can cause loss of efficiency in the system being monitored.

Perfmeter

One of the more useful tools is *perfmeter*. One useful way of using this tool is to bring up a set of windows each containing a *perfmeter*. Each window should monitor a major machine (system) within ECS. More information about this approach is given later. *Perfmeter* is an Open Windows XView tool. While it is supplied with Open Windows, it turns out that *perfmeter* will run in most UNIX windowing environments. The typical use is to monitor the CPU utilization of the main servers in the system. With experience an operator can recognize certain conditions within ECS as they arise. In general, this requires the operator to know approximately what is scheduled to happen within the ECS and when it should happen. Even without a detailed knowledge of the system's schedule, the operator should be able to spot potential problems via sudden increases or decreases of processor utilization.

Perfmeter has a number of parameters that control its operation. One of these is the sampling rate. Its default sampling rate is once every two seconds. This is probably too frequent for normal monitoring. A more typical rate is once every 15 seconds, although it is quite conceivable that an operator would want an even less frequent sampling. The trade-off is more frequent sampling adds somewhat to the system's load and shows more transient effects. The less frequent sampling averages out transients and makes it easier to see trends in system performance. In addition, it shows more of the recent performance history so that operator can compare current performance to recent performance in order to determine whether the system is reacting normally.

There are a couple of aspects of *perfmeter* than should be commented on. The first is that *perfmeter* normalizes the process utilization to the number of CPUs that are in the machine. As an example, 50% utilization on a machine that contains eight (8) CPUs means that an equivalent of four (4) CPUs are active, while 50% utilization on a machine with only two (2) CPUs, means that the equivalent of a single CPU is active. It should be noted that the normalization does not allow one to distinguish between multiple CPUs being partially active and one CPU being very busy.

The second aspect that needs comment is how *perfmeter* presents its graphs. The graphs are presented with time (unlabeled) on the x-axis. Increasing time is to the right, so the most recent samples are placed on the rightmost portion of the graph. Like the x-axis, the y-axis is not particularly labeled. The parameter being monitored is in the lower left. The baseline (lowest level) of the graph is always zero (0), regardless of the parameter being monitored. The maximum value of the graph is shown in the lower right hand corner. For parameters other than utilization *perfmeter* will rescale automatically depending on the range of data being presented.

With any performance monitoring tool, one wants to make sure that the tool is not interfering too much with the system being monitored. As a consequence, it is reasonable to inquire as to how *perfmeter* accomplishes its monitoring. Basically *perfmeter* accesses, via the proper protocol, the *rstatd* daemon on the computer(s) being monitored. The daemon returns the requested data. *Perfmeter* then formats the returned data into the various graphs. Two things flow from this. One is that the *rstatd* daemon must be running on each system that is to be monitored. In ECS this is usually the case, so it is not a particular worry. The second is that since the data collection is somewhat built into UNIX, the process is fairly efficient. A further observation is that if the computer running the *perfmeter(s)* is not one of the primary ECS machines, any load induced by producing the graphs is kept off the monitored machines.

On certain versions of UNIX the window control buttons are not present. This can cause one to wonder how to properly exit *perfmeter*. Normal termination is accomplished by placing the cursor in the active portion of the *perfmeter* window and typing the letter ‘q’.

Other *perfmeter* parameters allow the operator to monitor more than just the system utilization. These monitoring points can be displayed instead of or in addition to the utilization. The operator will have to decide whether any of the other monitoring points provides information that is useful in assessing the system’s health. The interested reader can consult the *man* pages on a Sun® system to find out about other monitoring parameters.

The following script is an example of using *perfmeter* to monitor the primary systems within a DAAC. This script samples every 15 seconds. With the size of windows that are specified in the script, between 1.5 to 2 hours of performance history is displayed.

```
 /usr/openwin/bin/perfmeter -geometry 300x80+844+0   -s 15 g0icg01 &
/usr/openwin/bin/perfmeter -geometry 300x80+844+108 -s 15 g0acg01 &
/usr/openwin/bin/perfmeter -geometry 300x80+844+216 -s 15 g0drg01 &
/usr/openwin/bin/perfmeter -geometry 300x80+844+324 -s 15 g0acs03 &
/usr/openwin/bin/perfmeter -geometry 300x80+844+432 -s 15 g0dis02 &
/usr/openwin/bin/perfmeter -geometry 300x80+844+540 -s 15 g0ins01 &
/usr/openwin/bin/perfmeter -geometry 300x80+844+648 -s 15 g0ins02 &
/usr/openwin/bin/perfmeter -geometry 300x80+534+0   -s 15 g0wkg01 &
```

```
/usr/openwin/bin/perfmeter -geometry 300x80+534+108 -s 15 g0mss10 &
/usr/openwin/bin/perfmeter -geometry 300x80+534+216 -s 15 g0pls01 &
/usr/openwin/bin/perfmeter -geometry 300x80+534+324 -s 15 g0pls02 &
/usr/openwin/bin/perfmeter -geometry 300x80+534+432 -s 15 g0sps06 &
/usr/openwin/bin/perfmeter -geometry 300x80+534+540 -s 15 g0spg01 &
/usr/openwin/bin/perfmeter -geometry 300x80+534+648 -s 15 g0spg07 &
```

MRTG (Multi Router Traffic Grapher)

The Multi Router Traffic Grapher (MRTG) is a tool to graphically monitor the traffic load on various network links. MRTG creates graphs that are embedded into web pages and are viewable via a typical web browser. In addition to a daily view, MRTG also creates graphs of the traffic over longer periods such as the last week, the last month, and the last year. MRTG accesses SNMP counters, so it is not limited to monitoring network traffic only, it is possible to monitor any SNMP variable desired. When MRTG is used, the web browser must be forced to reload the page regularly since MRTG has no mechanism for updating the graphs in real-time. MRTG is available the GNU General Public License. For further information, please consult the following web site:

http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html

Sar (System Activity Reporter)

Another tool that is useful for monitoring is the System Activity Report, known as *sar* because of the command which is used to invoke it. This tool does not directly present its data graphically, nor does it run remotely. The *sar* tool reports data only on the computer it was invoked from. Depending on the options that it was invoked with, it with either print its data in tabular form to the window, or it will write the data to a file specified by the user. An awkwardness with *sar* is that it will sample only for the number of intervals that was specified in the command. If the number of intervals is not specified it assumes a default of one (1). As a consequence on these limitations, *sar* is probably best used as a diagnostic tool, determining what is happening during relatively limited times on specific machines rather than a long term system monitor. The *sar* command should not have its sampling interval set to less than five (5) seconds, and a more typical range would be sampling every 30 seconds.

The *sar* command in interactive mode is:

```
sar [options] [interval] [number of samples]
```

The length of the sampling interval is specified in seconds. Some useful options of the *sar* command are *–d*, *–q*, *-r*, and *–u*. The *–d* option gives the activity of the disks connected to the system. The values reported include the average number of outstanding requests during the time period, the average time, in milliseconds, that transfer requests wait on the queue, and the portion of time the disk was

busy servicing requests. The *–q* option gives the number of processes in memory ready to run, the percentage of the time interval the run queue had processes waiting, the number of processes swapped out but ready to run, and the percentage of the time interval the swap queue had processing waiting. The *–r* option gives the system's amount of available memory (in pages) and the number of disk blocks available for swapping. The *–u* option is the default option and reports CPU utilization. The reported value is normalized to the number of processors (CPUs) in the computer system.

It is possible to do longer-term trend analysis using commands that are adjunct to the *sar* command. These commands are *sadc, sa1,* and *sa2.* These commands allow one to capture system activity over a long period, but at a relatively low time resolution (less frequent data samples). A typical sampling interval for *sadc* would be five (5) minutes or longer. Use of these commands is beyond the scope of this document. The interested user is directed to the *man* pages on *sar* and *sadc*.

Since graphical output is not directly available from the *sar* command, the typical analysis method is either to work with the raw numeric data directly or to transfer the data to a PC or Macintosh® system and plot the data using Excel®. On Sun® systems with certain terminals, the *sag* command can be used to create graphical representations of the data.

Smtstat (Station Management Table Status)

The station management table status command, *smtstat*, provides a fair amount of information on the FDDI-related data and station management tables. Unfortunately, it runs only on the SGI® computers and reports data only on the local FDDI port. Like several other of the SGI® tools, *smtstat* has a continuous mode (*i.e.*, it continuous updates the data at a fixed sampling frequency). Most of the information presented by the tool is a bit technical and presupposes a certain amount of knowledge about FDDI networks. However, there are three pieces of data that useful for determining the load and response of the FDDI. These are ring latency, ring load, and token latency.

The default sampling rate of *smtstat* is probably too frequent (too rapid) for monitoring a FDDI port for long periods of time. It is recommended that a sampling interval in the range of 5-to-30 seconds be used, unless there is a specific need to sample at a higher frequency. There are several different pages of information available via *smtstat*. The page that contains the latency and load data is page one (1). In addition, there are several modes in which the tool operates. The *D* (delta) mode is the one that is typically of the most interest to an operator.

The command for continuous sampling is:

smtstat –C [interval]

The letter *C* specified here is in upper case. The length of the sampling interval is specified in seconds. To exit *smtstat* when in continuous mode, just depress the *q* key. For further details on the *smtstat* command please consult the *man* pages.

Netstat (Network Status)

Versions of the network status command, *netstat*, exist on both the Sun® and SGI® systems. While the command runs on both systems, the SGI® version is a bit more useful and provides a continuous updating of the data. Like the *smtstat* and *sar* commands, *netstat* reports only on the ports of the system the command was run on. In continuous mode, the *netstat* command gives the number of protocol packets transferred on each of the network interfaces of the system being monitored.

Like the *smtstat* command the default sampling rate is probably too frequent (too rapid). It is recommended that a sampling interval in the range of 5-to-30 seconds be used. There are several different pages of information available via the *netstat* command. Page one (1) is probably the most useful page. In addition, there are several modes in which the tool operates. The *D* (delta) mode is the one that is typically of the most interest to an operator.

The command for continuous sampling is:

netstat –C [interval]

The letter *C* specified here is in upper case. The length of the sampling interval is specified in seconds. To exit *netstat* when in continuous mode, just depress the *q* key. For further details on the *netstat* command please consult the *man* pages.

Sysperf (AMASS Status)

Sysperf displays the status of the AMASS I/O activity. It is useful in determining the number of fnodes and cache blocks available. It can also be used to determine whether or not data is flowing onto archive tape. This latter can be a bit more difficult, since the AMASS has its own algorithm to determine when to move data from cache to tape. As consequence, the movement of a file to tape may decoupled from the time when the data is put into the AMASS cache.

The two parameters of most interest are the number of fnodes free and the number of cache blocks free. The total number of fnodes defines the maximum number of files that the AMASS file system can have open at any given time. The number of free fnodes, therefore, represents the remaining number of file transfers that can be initiated. If the number of free fnodes drops to zero, a condition referred to as fnode starvation occurs. While fnode starvation is not

desirable, it is not fatal. All that happens are additional transfers are held off until the AMASS can complete some of the outstanding file transfers. This does cause ECS to slow down until the AMASS catches up.

The AMASS cache is a disk area used to maintain open files and to stage reads and writes to and from the AMASS file system. The cache is used by AMASS as a sort of buffer to level its load. It also allows files that are in high demand to be fetched from the cache without the need to reread them from tape. The cache is divided up into blocks. A block is considered *dirty* if data (information) has been placed into it that has not yet been picked up by its end user (either a component of ECS or AMASS itself). It is possible to request enough information to be written to or read from the AMASS that the number of available (free) cache blocks drops to zero. As with fnode starvation, this is not a good situation but it is not fatal. Also like fnode starvation, it does cause to slow down.

There is a caveat with running *sysperf*. Some releases of the AMASS software do not report the correct data transfer rates if more than one *sysperf* is running. This does not affect the reporting of fnodes and cache blocks, just transfer rates. The *sysperf* command is:

sysperf –kc [interval]

The length of the sampling interval is specified in seconds.

Start_perf (performance data collection)

Start_perf is a group of scripts that collect various performance parameters from the various servers within ECS. It collects all of the parameters in a set of files and is, therefore, useful in documenting system performance during tests. A description of the features of *start_perf* is beyond the scope of this FAQ document and is described in a separate document.

- **What is an fnode?**

An fnode is roughly the AMASS analogue of an inode in UNIX. The total number of fnodes defines the maximum number of files that the AMASS file system can have open at any given time. The number of free fnodes represents the remaining number of file transfers that can be initiated. If the number of free fnodes drops to zero, fnode starvation occurs. While this is not desirable, it is not fatal. Additional transfers are held off until the AMASS can complete some of the outstanding file transfers. This will cause ECS to slow down until the AMASS catches up.

An fnode is in use while a file is being transferred. This means not just while a file is on the network, but also while it is waiting to be written onto or read from tape. The maximum number of fnodes interacts with different parts of the ECS system

in indirect, but concrete ways. As an example, as the maximum number of fnodes is raised, more file transfers can run in parallel. This has the effect of increasing the demand for cache blocks. What can happen, therefore, is that by avoiding fnode starvation, the bottleneck may be moved to lack of cache blocks or to network bandwidth demand. Since fnodes are in use while AMASS is transferring a file, the number of AMASS drives also enters into decisions about what value to use as the maximum number of fnodes.

The maximum number of fnodes available is an AMASS configuration parameter and should be changed only by those who are authorized to modify the AMASS operations.

- **What is a cache block and/or a dirty cache block?**

The AMASS cache is a disk area used by the AMASS system to maintain open files and to stage reads and writes to and from the AMASS file system. The cache is used by AMASS as sort of buffer in order to level the demand. It also allows files that are in high demand to be fetched from the cache without the need to reread them from tape. The cache is divided into blocks, which are the smallest chunk of cache that the AMASS system will allocate. A cache block is considered dirty if data (information) has been placed into it that has not yet been read by its end user (either a subsystem within ECS or AMASS itself, depending on the direction of file transfer). It is possible to request the transfer of enough information that the number of free (available/not dirty) cache blocks drops to zero. While this is not desirable, it is not fatal. Additional transfers are held off until the AMASS can complete some of the outstanding file transfers. This will cause ECS to slow down until the AMASS catches up.

The number of cache blocks is affected by a number of parameters, such as the total amount of cache available and the size of a cache block. It is indirectly affected by the total number of fnodes and AMASS drives available. Since they are used in file transfers, the number of cache blocks used interacts with the networks. As a consequence, only authorized individuals should change the total number of cache blocks.

- **What performance parameters can be monitored?**

Depending on the platform, many performance parameters are available. Some of the parameters available have been discussed in answer to the question of what monitoring tools are available. Presented below is a table giving various performance parameters and their sources. It is up to the user determine which of these parameters will be best for his situation.

Statistic	Description	Notes	Source
%rcache	Cache hit ratio on reads		sar –b
%wcache	Cache hit ratio on writes		sar –b
bread/s	Transfers per second of data from disk to system buffers		sar –b
bwrit/s	Transfers per second of data to disk from system buffers		sar –b
lread/s	Transfers per second from system buffers to user memory		sar –b
lwrit/s	Transfers per second to system buffers from user memory		sar –b
pread/s	Read transfers using raw (physical) device mechanism		sar –b
pwrit/s	Write transfers using raw (physical) device mechanism		sar –b
%busy	Portion of time device was busy servicing a transfer request		sar –d
avque	Average number of requests outstanding during %busy time		sar –d
avseek	Time, in milliseconds, for average seek	5	sar –d
avserv	Average time, in milliseconds, to be serviced (which for disks includes seek, rotational latency and data transfer times)	1	sar –d
avwait	Average time, in milliseconds, that transfer requests wait on queue	1	sar –d
blks/s	Number of bytes transferred in 512-byte units		sar –d
r+w/s	Number of data transfers from or to device	1	sar –d
read/s	Number of read transfers from device	5	sar –d
write/s	Number of write transfers to device	5	sar –d
atch/s	Page faults per second satisfied by reclaiming a page currently in memory	5	sar –p
cache/s	Address translation fault page reclaimed from page cache	1,6	sar –p
cpyw/s	Protection fault on shared copy-on-write page	1,7	sar –p
dfill/s	Address translation fault on demand fill or demand zero page	1,6	sar –p
pflt/s	Page faults from protection errors per second or “copy-on-writes”		sar –p
pgfil/s	Address translation fault page reclaimed from file system	1,6	sar –p
pgin/s	Page-in requests per second	5	sar –p
ppgin/s	Pages paged-in per second	5	sar –p
pswp/s	Address translation fault page reclaimed from swap space	1,6	sar –p
rclm/s	Pages reclaimed by paging daemon	1	sar –p
slock/s	Faults per second caused by software lock requests requiring physical I/O	5	sar –p
steal/s	Protection fault on unshared writable page	1,7	sar –p
vflt/s	Address translation page faults per second (valid page not in mem.)		sar –p
%runocc	Percentage of time interval the run queue has had processes waiting		sar –q
%swpocc	Percentage of time interval swap queue has had processes waiting	1	sar –q
runq-sz	Average number of processes waiting in run queue because no processor is available.		sar –q
swpq-sz	Average number of processes runnable but swapped out	1	sar –q
freemem	Average number of memory pages available		sar –r
freeswap	Number of disk blocks available for process swapping		sar –r
%idle	Percentage of time interval CPUs spent totally idle	2	sar –u
%intr	Percentage of time interval CPUs spent servicing interrupts	2,3	sar –u
%sbrk	Percentage of time interval idle waiting for system memory	2,3	sar –u
%sys	Percentage of time interval CPUs spent in the system mode	2	sar –u
%usr	Percentage of time interval CPUs spent in the user mode	2	sar –u
%wfif	Percentage of wio time spent waiting while graphs pipe is too full	3,4	sar –u
%wfs	Percentage of wio time spent waiting for File System I/O	3,4	sar –u

%wgsw	Percentage of wio time spent waiting for graphics context switch	3,4	sar –u
%wio	Percentage of time interval CPUs spent waiting on I/O operations	2	sar –u
%wphy	Percentage of wio time spent waiting for phys. I/O other than swap	3,4	sar –u
%wswp	Percentage of wio time spent waiting for swap I/O to complete	3,4	sar –u
bswin/s	Number of 512-byte blocks transferred from swap space		sar –w
bswot/s	Number of 512-byte blocks transferred to swap space		sar –w
pswch/s	Number of process switches	5	sar –w
pswpout/s	Number of process swapouts	3	sar –w
swpin/s	Number of transfers from swap space		sar –w
swpot/s	Number of transfers to swap space		sar –w
fnodes	The number of potentially open files in the AMASS file system		sysperf

Notes:
1. This parameter is not available on Sun® systems.
2. The sum of these parameters should be 100%.
3. This parameter is available only on SGI IRIX® systems.
4. The sum of these parameters should be 100%.
5. This parameter is not available on SGI IRIX® systems.
6. This parameter is a subset of vflt/s.
7. This parameter is a subset of pflt/s.

- **Is there a specific setup that is recommended for operations monitoring?**

It would seem that the configuration that provides the most information while having a fairly small impact on system resources is to set up two (2) terminals. A server that is lightly loaded, such as a xterm host, should drive these two terminals. These two terminals should close be enough to the operations controller that he can monitor the information while performing his normal duties. Depending on availability of terminals, the number and size of the windows used, and the preferences of the operator, a single terminal may be used instead of two.

On one of these terminals, a set of *perfmeters* can be brought up, one per major system. The sample set of *perfmeters* that was previously given is an example of this. The sampling interval for these *perfmeters* should be 15 seconds. Longer sampling intervals can be used if the operations staff so desires, but it not recommended that an interval of less than 15 seconds be used for an extended period of time. The typical parameter monitored by the *perfmeters* is system utilization, however, the operations staff may find that additional parameters provide useful information.

Also on this terminal, a *netstat* on the archive machine(s) should be brought up in continuous mode. This will allow monitoring of the HiPPI traffic to and from PDPS. It will also allow the monitoring of ingested data being sent to the archive. If useful, a *netstat* can be brought on the APC server and placed on this terminal.

This latter display will show the ftp push/pull traffic. Like the *perfmeters*, the sampling interval of any *netstats* should be set to 15 seconds or longer.

On the other terminal a web browser can be brought up and MRTG placed on it. The sampling interval of MRTG cannot be changed in a convenient fashion. In addition, current releases of MRTG do not automatically update, so the operator will need to reload the web page occasionally to get updated displays. If desired a *sysperf* can be brought up in order to monitor the general performance of the AMASS. The default sampling interval for *sysperf* is 1 minute (60 seconds). It seems reasonable to stay with this sampling rate.

- **Is there a recommendation for long term (trend) monitoring?**

Long term monitoring can be used to detect some problems and to predict the need for additional system resources or the need to reallocate existing resources. For this monitoring it is recommended that *sadc* (a part of the system activity report, *sar*, package) collecting all parameters be used. The sampling interval for this should be set to 5 minutes (300 seconds) or longer. Because of the default file naming conventions of *sadc* the files containing the performance data will have to be regularly saved off under different names. For more information the *man* page for *sadc* and *sar* should be consulted. The MRTG program automatically keeps long term trend data, admittedly at lower resolution.

The traditional problem with the *sar* command data is that there is no graphical method of presenting the data (trend data is easier to analyze if it is presented in graphical form). The typical analysis method is either to work with the raw numeric data directly or to transfer the data to a PC or Macintosh® system and plot the data using Excel®. On Sun® systems with certain terminals, the *sag* command can be used to create graphical representations of the data. MRTG contains its own plotting routines, so graphical output from it is not an issue.

- **What tools does AMASS provide for monitoring?**

AMASS provides a set of monitoring tools under the path /usr/amass/bin. Depending on permissions established at installation, access to the tools may be limited only to amass user accounts. There are three useful monitoring tools. Sysperf (as discussed already) shows the current status of the drives, the number of dirty blocks in the cache, the number of outstanding write requests, and the i/o load on each drive. quedisplay shows the queued requests, whether they are read or write, the tape the request is queued against, the drive status and which tapes are in the drive. vollist lists the media volumes in the library and gives the status for each one. vollist can be used to determine if a media volume has been taken off-line.

Knowledge of how AMASS performs its functions is necessary for understanding what the output of the above tools really means. AMASS services writes over

reads in accessing the cache, but services reads over writes in accessing data on tape. If all of the data is being put into the same volume group and scattered writes is not enabled, then the all of the data is being written to one tape. Requests for that data will then cause delays in the writing of additional data to the tape. If the size of the files is small, then the file being written can be put completely in cache and no discernable system delay will result. However, if the files are large enough, the number of dirty blocks per write request can be filled. This will cause the write process to hang until some of the dirty blocks can be written to tape. In this situation a large number of reads against the tape will cause the write process to hang for long periods of time. This situation was observed recently at EDC. Given time, the situation will resolve itself. Stopping AMASS by rebooting the system should be avoided since it leads to database corruption.

- **What tools does Sybase provide for monitoring?**

Several tools are available for monitoring sybase. All tools require system administratior (sa) privileges on the Sybase Server. The DBA should be able to provide assistance with giving information on state of the Sybase Server, or provide access to the tools to a trusted user.

<u>ECS Scripts</u>
EcDdmMonitorSegment – monitors database segment space allocation
EcDdmMonitorServer – monitors Sybase server performance
EcDdmUserCounts – monitors user connections

In addition, SQL Monitor Server/Client and DBVision provide performance data via a graphical user interface (GUI). DBVision provides email notification capabilities when performance thresholds have be passed by the Sybase Server.

Several SQL scripts install as part of the database build. These scripts attempt to prevent cessation of database activity due to full database segments utilization. These scripts issue warnings (to the Sybase Server error log) when the database segment reaches a given percentage of utilization, or, for log segments, dumps the database transaction log when the threshold is crossed (20% - warning, 15% - dump, 10% -dump). Data segment scripts issue a warning for the following thresholds threshold (20% - warning, 15% - warning, 10% - warning, 5% - warning).

- **What system settings can / should be monitored, and what are these settings for?**

Transaction Log Segment

Monitor the Sybase error log for warning and transaction dumps due to thresholds. If the frequency of dumps do to thresholds is increasing, the size of the transaction log should be increased.

Additionally, the transaction dump device should be monitored for space. If the dump device fills up, transaction dumps can not complete, and this may eventually lock up the application because database transactions can not be completed.

The size of the transaction logs can be monitored using: 1) DBVision, 2)ECS script EcDdmMonitorSegment.

Transaction dumps due to thresholds should be kept to a minimum. If they are occuring, the size of the log segment should be increased, or the period for database transactions dumps should be shortened. All ECS databases transaction dumps should occur at the same scheduled intervals, since a restore of one subsystem database may require a database restore of another subsystem.

The growth rate of the transaction log should be monitored. This can be accomplished using: 1) DBVision, or 2) ECS script EcDdmMonitorServer.

Data Segments
Additionally, the Sybase error log should monitored for warning messages related to space allocation of the data segments (as opposed to log segments). Full data segment utilization could also halt all of storage management database processing. If a data segment starts to fill up, the System Administrator has two choices: a) increase the size of the segment, or b) reduce the amount of data stored in the database.

Data segments can be monitored via: 1) DBVision, 2) ECS script EcDdmMonitorSegment.

User Connections

The number of user connections needs to be monitored periodically. If the number of user connections exceed the current Sybase server configuration parameter, applications requesting the additional connection will not be able to connect to the Sybase Server. If the number of user connections is being exceeded, either the numbers of threads needs to be reduces, or the Sybase

parameter needs to be increased. The number of connections should be set to handle all connections required *for all databases in all modes* for the given Sybase Server.

User connections can be monitored using 1) DBVision, 2) ECS script EcDdmUserCount.

Current Baseline Sybase Server Configurations	
Sybase Server	**Number of User Connections**
e0acg01	255
e0icg01	140
g0icg01	145
g0acg01	250
l0acg01	250
loicg01	150
n0acg01	250

Sybase Server Performance

The following general performance areas should be monitored.

CPU Utilization – should stay below 90% most of the time.
Cache Hit Ration – should stay above 85% most of the time.
Lock Management – should be little or no lock contention.
Disk Utilization – disk utilization should not should show device contention, and I/O should be distributed among the database devices.

In general, Sybase configuration parameters related to these areas should be left to their default, unless performance problems are being experience by the subsystem, and monitoring of the Sybase server indicates a problem in any one of the above areas. Actions to be taken are detailed in the Sybase manual, SQL Server Performance and Tuning Guide.

Sybase performance can be monitored using: 1) SQL Monitor Server/Client, 2) DBVision, 3) ECS script EcDdmMonitorServer.